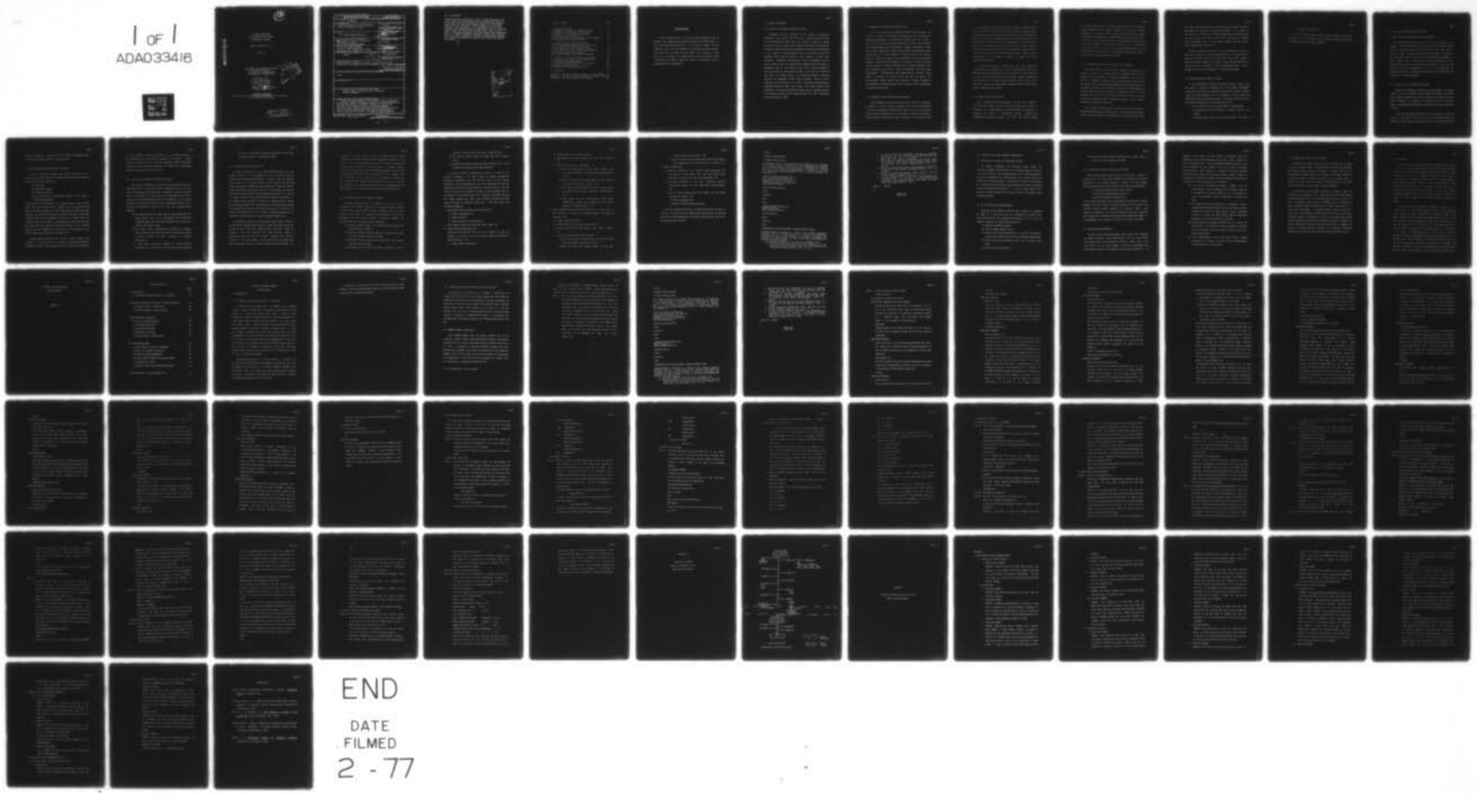
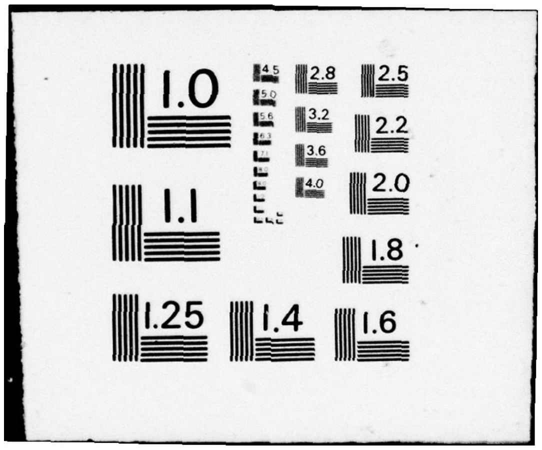



1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(12)

A MODEL INTERFACE FOR THE DECISION AIDING INFORMATION SYSTEM

Ralph M. Mitchell Jr.

76-09-11

Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, Pennsylvania

408 757

[illegible]

Prepared for the
Office of Naval Research
Information Systems
Arlington, Va. 22217
under
Contract No. ~~N0014~~ N00014-75-C-0440
Project No. NR049-360

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 14 76-09-11	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 A Model Interface for the Decision Aiding Information System.		5. TYPE OF REPORT & PERIOD COVERED 9 Interim rept.
		6. PERFORMING ORG. REPORT NUMBER 76-09-11
7. AUTHOR(s) 10 Ralph M. Mitchell, Jr		8. CONTRACT OR GRANT NUMBER(s) ~~N0014~~-75-C-0440 15 N00014-
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Decision Sciences The Wharton School University of Pennsylvania		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-049-360
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Code 437, 800 North Quincy Avenue Arlington, Va. 22217		12. REPORT DATE 11 Sep[redacted]976
		13. NUMBER OF PAGES 66
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (same) 12 71p.		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)
Reproduction in whole or in part is permitted for any purposes of the United States Government

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
(same)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Models, Model Interface, User Interface, Decision Making

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Under the sponsorship of the Office of Naval Research, the Decision Aiding Information System is being developed at the Wharton School to provide a vehicle to assist the tactical level decision maker in bringing together available data, models and programs and through and active role in user/system interactions improve the manager's decision making activities. In today's complex systems, → over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408 757 ✓

mt

20. (continued)

models have the capacity to play an ever-expanding role in the decision environment, but a system was required that would provide the decision maker with information which will (a) lead the decision maker to the proper model for the situation, (b) automatically find and transfer the required data to the model, and (c) transfer results back to the decision maker in an easy to use form. This paper explains the development and implementation of a Model Interface Program to meet these requirements in order to provide a bridge to enable the manager to integrate mathematical models into his own decision processes.

Table of Contents

ACKNOWLEDGEMENT

I thank Professor Howard L. Morgan, my thesis supervisor, for his patience and understanding guidance throughout the entire project. I also thank Professor David Ness for his revealing insights on many important issues. For the demanding task of placing the draft copy of this thesis into its present form, my thanks are extended to David Root. And as always, my thanks must go to my wife, Shirley, whose encouragement and support during the times of frustration were a constant source of inspiration.

1.0 PROBLEM ENVIRONMENT

1.1 Management's Dilemma-an Explanatory Model

Management has been described as the process of converting information into action with the actual conversion process called Decision Making.[Paik, 1973] With the computer information systems available today, the wider span of control and inherent multiplication of complex interactions between variables has reduced our understanding of the dynamics of these processes to such a point that our intuitive judgement can no longer be trusted in many instances to make valid decisions. Overwhelmed with the sheer volume of information that can now be brought to bear with sophisticated computer retrieval systems, a monumental task of the decision maker (DM) is simply defining the relevant variables to the decision at hand. It is from this "intuitive gap" that the related fields of operations management, operations research and management science have developed. Russell Ackoff summarizes the disciplines' 20 year's of research and management's continued frustration with these words, "one cannot specify what information is required for decision making until an explanatory model of the decision process and the system involved has been constructed and tested."[Ackoff, 1967]

1.2 Emergence Of Modeling As A Decision Aid

It is this emphasis on "an explanatory model" that has become the fulcrum to enable us to reduce the complexity of the real world into manageable proportions. The mathematical methods that have found a place in the literature have been voluminous and include, among others, linear programming, dynamic programming, integer programming, simple and multiple regression analysis, exponential smoothing, Monte Carlo simulation et. al. Entire subsets of problem types have emerged: resource allocation, product line determination, distribution-location analysis, transportation programming, risk analysis, decision analysis (utility theory, stochastic decision trees, inventory problems, short term scheduling and control, demand forecasting, and long range forecasting). Mathematically and computationally feasible, these models of decision and control allow the user to observe the relationships between factors and objectives, to obtain a measure of effectiveness to evaluate decisions and to predict future consequences of selected alternatives.

1.3 Applicability To The Task Force Environment

This tremendous volume of literature has not met with unqualified acceptance in actual practice, however, due in part because managers are not yet convinced that mathematical models can fully and accurately describe what is happening in the "real world". However, the weight of evidence seems to indicate that these techniques, intelligently used, can be of immense benefit in situations of complexity and uncertainty. This is exactly the position facing the Navy Task Force Commander whose already professionally demanding requirements for effective command and control of the myriad interdependent task force elements has now been exacerbated by a multi-level threat environment with a far greater kill range and effectiveness than ever faced before. The reduced response time and increased payload of enemy strike threats no longer afford the luxury of recovering from less than optimal decisions reached without consideration of all pertinent variables of present task force capabilities/limitations.

In looking for methods to enhance the task force commander's effectiveness, this large body of literature on mathematical models for desision making provided the impetus for the Office of Naval Research to fund the Operational Decision Aiding Project (ODAP). Emphasizing the utilization of existing techniques which could provide better operational decisions, ONR assigned independent applications-oriented tasks to numerous subcontractors.

1.4 DAISY--A Decision Facilitator

But, it was perceived that providing the user with interactive tools was not enough. Tactical level decision makers had to have, in the words of Dr. Howard MORGAN of the Wharton School, "the ability to integrate the latest in information systems, modeling, and probabilistic estimation techniques into their own decision making activities."[Morgan, pg 3] It was for this purpose that the Decision Aiding Information System (DAISY) emerged at the Wharton School's Department of Decision Sciences under the direction of Dr. MORGAN. Throughout its development, the emphasis has been on DAISY'S role as a facilitator that will allow the user to bring together available data, models, and programs and through an active role in user/system interactions assist the user in his decision making activities. The common theme since inception is that the system should provide aids for the DM and not to simply automate decisions.

1.5 The Cornerstones - Decisions, Models, Data, Triggers

Three basic levels of decisions exist in DAISY. The generic level contains a series of prompts that form the framework for the user's approach to decision making. In using this framework, the user will develop a "specific" decision for the generic decision type at hand. For instance in the task force environment a generic decision might be refueling operations and the specific decision to refuel destroyers. The final link in this chain is the actual decision to refuel the USS DANIELS, and this will then become a decision "instance" with the inclusion of appropriate input parameters (gallons of fuel needed, distance from refueling ships etc.)

Data is defined as both global and local, with local data used for scratchpad purposes and/or personal notes. The global data may contain any data which is accessible to DAISY and, if not already in the user's data base, becomes a candidate for addition to the system if the user determines it is relevant to his decision processes. It is added to the system by providing the information required by the Generic Data Item format (data file, access path, data format etc.). Another feature described later is that DAISY accesses the data when a user-designated model requires it.

Triggers indicate the occurrence of an event which may or may not cause the initiation of some pre-defined action. The event may be a decision to launch a carrier air strike which would then trigger a model to calculate flight time to target. Another example could be the receipt of data on a casualty to a ship's engines which may trigger a message to the task force commander to slow formation speed.

1.6 The Task Defined--A Model Interface

It is the remaining cornerstone of DAISY, the model entity, that the author is directly concerned with in this paper. With ODAP's interest in the potential of mathematical models for decision making, "DAISY has been designed from the beginning to make access to sophisticated models as easy as possible. This means that the decision maker must be provided with information which will:

(a) lead him or her to the proper model for the situation

(b) automatically find and transfer the required data to the model

(c) transfer results back to the decision maker in an easy to use form." [Morgan, pg 7]

It was the author's task to develop a model interface that would accomplish these requirements and it is this development process that is the subject of the succeeding chapters.

2.0 A MODEL INTERFACE--THEORY AND DESIGN

2.1 Basic Requirements--Generality Vs Specificity

The basic problem facing the designer of a model interface is to produce a framework general enough to accomodate the infinite variety of models potentially available to the DM while at the same time be specific enough to handle the particular requirements of each model. The ultimate goal was to enable the DM to use any model available by a simple command such as "RUN MODEL Air Strike Optimization", and the model interface would then automatically be invoked to carry out the necessary data/file manipulations as well as be responsible for prompting the DM for any information required from the user during the execution of the model selected.

2.2 The Search For An "Equalizing Function"

Taking into consideration these basic requirements, the design approach used had to embrace a global-to-specific viewpoint that would allow a smooth integration of diverse, external model entities into the DAISY environment by creating some "equalizing function" that would enable a standard interface program to manipulate any model selected by the user.

To find this "equalizing function", it was necessary to review the nature of the model entity itself. HITCH and McKEAN remind us that models are simply "abstract representations of reality which help us to perceive significant relations in the real world, to manipulate them, and thereby to predict others. "[Hitch, pg 119]

2.3 Model Manipulation--The "Black Box" Concept

If the relationships between real world events are to be represented, one would expect that every model builder has to construct his model within the framework of:

(1) data inputs

(2) information outputs

(3) structural elements

(4) a logic system of relationships usually in the form of mathematical formulas

Perhaps then, for our purposes, if the model builder could provide DAISY with the specific format required for the data inputs and resultant outputs, coupled with a brief description of basic structural elements, the specific relationships detailed in the body of each model could be treated by the programming interface as a "black box." The interface would then gather the necessary data values from the data base and/or the user, present this data to the model in the required format and then assemble the output data for presentation to the decision maker upon completion of the model run.

Having assumed this "black box" view was a viable concept, what remained to be outlined was the method of capturing the required information elements for each model available in the DAISY environment.

If these elements could be captured on a one-time basis using a standardized format as each model was added to the system, a single interface program could then select the appropriate model's descriptive module and proceed with model execution. The sought after "equalizing function" would then have been attained.

2.4 The Interface Keystone--a "Descriptive Module"

From a global perspective, this proposed system seemed to meet the requirements for a generalized model interface, but to fill in the specific characteristics of the system components, further questions had to be resolved. The first question to be addressed, and one that would have far-reaching implications for the remainder of the system design was to determine the nature of this "descriptive module." It was felt that the descriptive module had to meet the following basic criteria:

(1) The composition of this module must be simple enough that the model builder will not be discouraged from introducing subsequent models into the DAISY environment that could be of value to the decision maker;

(2) The module should comprehensively describe the relevant input, output and structural information concerning the model in order to allow the model interface to function effectively;

(3) A brief model description followed by a more detailed amplification of the model's intended use should be included

to allow the DM to make intelligent selections of model types with applicability to the problem at hand.

2.5 The Semi-structured File Concept

If these criteria were to be effectively/efficiently met, the vehicle chosen for the accumulation of this data should have the flexibility to accomodate introduction of textual data of variable field lengths in a loosely defined format while at the same time providing a method of segmenting the descriptions of different elements (input data types, output data types etc.). Fortunately, this area of loosely structured character string/textual data has been the subject of ongoing research conducted by the Wharton School's David NESS. In his working paper entitled "A Family of Systems which Process Semi-Structured Information", NESS states that semi-structured problems "are characterized by the fact that while little of the information is directly structurable into well defined fields, nevertheless it is possible to provide some kind of useful structure."[Ness, pg 4]

Of particular interest to the descriptive module notion is that if the information supplied cannot easily be forced into highly structured fields, we could treat the semi-structured information simply as strings of text with delimiters that bound different fields of information. NESS calls this the "escape" character and uses the up-arrow (^) for this purpose. The up-arrow alerts the program that the next character will be a "control" character that will signify the purpose of the next string of text. In essence then, the program searches for an escape-control character combination to begin the file (^A in our case) and scans the text until it finds an escape-control character pair of interest. It then processes the ensuing character string until another up-arrow is encountered signifying the end of the current character string. The search process is then repeated until the end-of-file pair (^L in our case) is encountered to terminate the file search process. Armed with this concept, we now have a vehicle that provides the flexibility required within a straightforward, but highly effective and consistent package--the semi-structured file.

2.6 The Model Parameter File (MPF)--a Framework

What model parameters must be included in this file to meet requirement (2)'s effective functioning of the model interface criteria for a descriptive module, which we shall henceforth call the Model Parameter File (MPF)? It appeared that the minimal set of critical parameters that would allow effective interaction between the user-model-data base interface would be composed of:

(1) A quick, effective method of differentiating between each model available to DAISY;

(2) A generic variable definition and generic name for each input and output variable used by the model;

(3) The specific format required by the model for each generic input and output variable;

(4) The general type of input and output variables that are

available from the user's data base (a subset of (3));

(5) The specific storage format for these data base variable types;

(6) Information on whether these data base variables are to be automatically updated after model execution;

In finding a method of representing a solution to each of the required parameters, the basic tenets of logical consistency, simplicity, readability and for the first time in our systems development effort-programmability had to be constantly utilized when considering the format for each parameter. The first step in the MPF creation process was to determine the order in which parameters were to be arranged in the sequential MPF. The structure instituted follows the thought pattern one would tend to follow in creating the model itself (the reader is invited to follow FIG. 2-1's MPF during the following discussion):

1. Identify the specific model we're interested in:
   a. Numeric designation (~S)
   b. Model Title (~T)
   c. Actual location of the model (~f)
2. Brief summary of what this particular model "does" (~U)
3. Input Variable Characteristics--

   Now that we've identified the file we're looking for and the purpose of the model, we want to know what inputs are required for model execution. (~V)

   a. Input variable definitions

b. Variable name for each input required

c. What format does the model require for each input variable? (^g)

d. Can the data base be of assistance?

(1) List the variable names of those input variable types that are available in the user's data base (subset of 3b) (^h)

(2) What is the data base storage format for each variable type identified in 3.d.(1) above? (^i)

(3) Will values of model inputs received from the data base be modified during model execution? (YES/NO answer) (^j)

(4) Will these data base variables have their values automatically updated upon completion of the model's execution? (YES/NO answer). (^k)

4. Output variable characteristics--we now skip to the "other side of the blackbox" to identify the resultant outputs of the model we selected. (^W)

a. Output variable definition

b. Variable name for each output value produced

c. In what format does the model produce each output variable? (^l)

d. Is the data base affected by the outcome?

(1) List the type output variables that are contained in the data base (always a subset of 4.b) (^m)

(2) What is the data base storage format for each type

output variable listed above? (^n)

(3) Will these data base variables have their values updated upon completion of the model execution? (YES/NO) (^o)

5. Model Administration

a. What was the source of the original model development and introduction to the system? (i.e. who could supply us with additional information on model characteristics?) (^X)

b. Detailed description of the model explaining specific relationships germane to the formulation of model outputs. (^O)

c. When was the last change made to this model and the person executing the change? (^M)

(1) Date---day/month/year

(2) Name of person making modification

With this framework completed, the remaining question centered on creating the Model Interface Program which would extract the required data from the MPF and perform the required interaction with the DM for the specific model selected.

```
^A^S412

^TECONOMIC ORDER QUANTITY

^f1:MOD412.DAT[4010,104]

^UIN GENERAL NOTATION, THIS INVENTORY DECISION MODEL MAY  BE  EXPRESSED
AS  T = F(Q,R), WHERE "T" DENOTES THE TOTAL INVENTORY-RELATED COSTS FOR
A GIVEN PLANNING PERIOD, "Q" IS THE QUANTITY  REPLENISHED  OR  ORDERED,
AND  "R"  IS  THE TIMING OF THE REPLENISHMENT.  THE DECISION IS TO FIND
THE VALUES OF THE DECISION VARIABLES "Q" AND  "R"  WHICH  MINIMIZE  THE
TOTAL COST "T".

^VCOST OF ORDERING PER ORDER (C);
TOTAL ANNUAL REQUIREMENT FOR ITEM (D);
DEMAND RATE PER UNIT OF TIME (E);
CARRYING COST PER UNIT PER YEAR (K);
LEAD TIME PER ORDER (L);

^gF8.2;F5.0;F5.0;I3;F3.0;

^hC;K;

^iI5;F5.2;

^jYES

^kYES

^WECONOMIC ORDER QUANTITY (Q);
REORDER POINT (R);
TOTAL INVENTORY COST (T);

^lF8.0;F5.0;F7.2;

^mQ;R;

^nI5;F5.0;

^oYES

^XDEPARTMENT OF DECISION SCIENCES, WHARTON GRADUATE SCHOOL

^OBROADLY DEFINED, INVENTORY IS  A  STOCK  OF  ANY  ECONOMIC  RESOURCES
REMAINING  IDLE  AT  A  GIVEN POINT OF TIME, INCLUDING SUCH NONPHYSICAL
INVENTORIES AS LEVEL OF CASH, INVENTORY  OF  ACCOUNTS  RECEIVABLE,  AND
INVENTORY  OF  HUMAN  TALENTS  AS  WELL AS MORE FAMILIAR INVENTORIES OF
PHYSICAL GOODS AND MATERIALS.
     AS A SIMPLISTIC EXAMPLE IN THE TASK FORCE ENVIRONMENT, LET:
     1.  THE TOTAL DEMAND OF THE ITEM FOR THE NEXT PERIOD (D),  BE  THE
           ANTICIPATED TOTAL OF FIGHTER PLANES LOST IN THE SEIZURE OF AN
           ISLAND OVER A TEN DAY PERIOD.  (I.E.  D = 30)
```

```
2.  THE "CARRYING COST" FOR MAINTAINING ONE UNIT OF INVENTORY
     (PLANE) FOR TEN DAYS INCLUDES FUEL, REPLACEMENT PARTS, AND
     MAINTENANCE COSTS.  (I.E.  K = $20,000)
3.   KNOWN COST OF ORDERING REPLACEMENTS FROM NEAREST CONUS
     TRANSPORTATION POINT INCLUDES COST OF SURFACE/AIR FUEL COST,
     AND PERSONNEL COSTS DURING TRANSPORTATION.  (I.E.  C =
     $5,000)
4.  THERE IS A FOUR DAY LEAD TIME FOR TOTAL TRANSIT TIME FROM U.S.
     TO ARRIVAL OF PLANE ONBOARD THE AIRCRAFT CARRIER.  (I.E.  L =
     4 DAYS)
5.  THE RATE OF CASUALTY ACCUMULATION (USAGE) OVER THE TEN DAY
     PERIOD AVERAGES 3 PLANES PER DAY.  (I.E.  E = 3)
6.  IF USER PROVIDED DATABASE VARIABLES WITH EQUIVALENT AND
     APPLICABLE NAMES (C = DAT301), (K = DAT308) AND TYPED IN
     VALUES FOR D, E, L MENTIONED ABOVE, THEN MODEL 412 WOULD
     PROVIDE THE ANSWERS FOR Q, R, AND T.

^M25.8.75   ;MITCHE

^L
```

Figure 2-1
Sample MPF

3.0 CREATION OF THE MODEL INTERFACE PROGRAM (MIP)

3.1 Building The Interface--an Integrated Process

The design, development, and subsequent coding effort for successful programs has developed its own extensive body of literature in the last 15 years. The top-down-approach, the bottom-up-approach, structured programming and modularization have become part of the jargon of the software industry. Underlying each method is the concept of a logical flow that enables the program (and the reader) to proceed through the algorithmic structure in a step by step fashion that produces a smooth transition from module to module in creating a final integrated process.

3.2 The Functional Design Requirements

Following the development of the two basic concepts of treating the model as a black box and the construction of the MPF as our "equalizing function", the integrated final process for the MIP was created to parallel the logical flow pattern of:

(1) identify the model selected;

(2) read the model parameter file;

(3) interact with DM for modifications to generic definitions, names and/or insertion of any desired data base designations;

(4) obtain required input variable values from DM and/or data base;

(5) execute the selected model;

(6) provide the output values, along with input values used to the DM in a User's Summary File (USF).

3.3 A Three Phase Design - The Black Box Extension

Using these principle steps, the next task was then a matter of breaking down the "micro-steps" required to accomplish the six major functions for the proposed interface. However the first process involved a separation of these functions into three major components:

(1) The Pre-execution Phase; steps (1)-(4);

(2) the model execution phase-step (5);

(3) The Post-Execution Phase-step (6).

Each of these three phases were then designed as separate and distinct programs that would "pull in" the next program entity into core as the currently running program completed its execution phase. This separation emphasizes the black box concept and enables the programs that accomplish the required data manipulations (RMODEL and RETURN) to be independent of the location or content of the model file itself.

3.4 Emphasis On Modularization

The next step in systems development then evolved into isolating the logical processes within each program to carry out the six major steps required for a functioning model interface. Rather than to go into details of the actual program, a schematic is presented of the major FORTRAN subroutines and MACRO routine linkages (Appendix B). In

Appendix C the purpose of each routine is presented within the framework of the 3 major components and six basic steps required for the MIP discussed above. Since the FORTRAN language was chosen as the source language for the interface due to its almost universal transferrability, this emphasis on isolating the logical processes through extensive usage of the subroutine concept created a number of advantages that have paid dividends during development and should prove invaluable for future expansion of the interface:

(1) In both the pre-execution program (RMODEL) and the post-execution program (RETURN), the main routine is used as a "scheduler"--as the next process is to be performed, it calls the applicable module (subroutine) to perform that task.

(2) The segmentation of each process into a subroutine context, allows similar processes carried out on both input and output variables to share the same general purpose processing routines by having the MAIN routine pass the applicable arguments for the current process of interest.

(3) Future expansion of interface functions can be affected by modification of the specific process of interest or addition of a new subroutine called from either the MAIN routine or an existing subroutine.

(4) The "modularization" effect facilitates future program maintenance by keeping the coding in easily understandable blocks independent of one another.

3.5 The Model Interface--a Working Reality

Following these design precepts the Model Interface was brought "on line" in planned stages. January 1976 saw the system testing begin for handling models written in FORTRAN. February 1976 saw the commencement of systems testing of the bridge to the models resident in the APL environment. By March, the system was an integral part of DAISY and passed its first external viewing during the 1 April 1976 visit by members of the ONR project steering committee.

At the present time the model interface has already proven its versatility by operating with a wide range of model types including forecasting models such as an exponential smoothing model written in APL, a regression model written in FORTRAN as well as a probability based optimization model concerned with air strike capabilities in a task force environment. It appears, based on initial response from users, that the development's concentration on the logical flow pattern processes within a framework of the black box and three phase design concept has provided DAISY with an impressive facility to advance its goal of providing the decision maker with "the ability to integrate the latest in information systems, modeling, and probability estimation techniques into their own decision making activities."[Morgan, pg 3]

4.0 CONCLUSION

With the ever-increasing complexity of the management environment of the future, the effective manager will be forced to place greater reliance on the scientific-mathematical-quantitative methods in order to enhance his ability to discover, analyze and test relations contained within an expanding set of diverse factors abstracted from a given problem situation. However, if the decision maker is to make significant and effective use of available decision aids such as mathematical modeling, they must be available in a form that is readily understood, adaptable to the decision maker's "specific problem" and capable of being integrated in the regular decision process of the manager.

DAISY has explicitly recognized the importance of modeling as an integral part of the decision making process and the Model Interface may be viewed as a vehicle through which mathematical models could become dynamic tools rather than operational curiosities. It is hoped that this Model Interface Program as outlined from the user's point of view in Appendix A, will inspire the use of available models in the user's decision process and spark a renewed interest in and awareness of the power of mathematical models. For then, "The very effort of applying the formal methods explicitly and consciously would force the decision maker and his analysts to seek rigorously and creatively new options for achieving the same end and to question and re-examine the objectives themselves and the corresponding measures of effectiveness employed. Such certainly would be a major contribution."[Paik, pg 5]

Appendix A

Table of Contents

THE MODEL INTERFACE PROGRAM

A USER'S MANUAL

1.0 INTRODUCTION

1.1 The Model Interface Program (MIP) - An Overview

The Model Interface Program (MIP) is a FORTRAN source language software package developed for the Decision Aiding Information System (DAISY). It allows the decision maker (DM) to readily access mathematical models previously entered into the DAISY environment by a simple "RUN" command. The MIP will then determine what input values are needed, whether they're obtainable from the data base or the DM, and what outputs are generated. The MIP then allows the DM to change any variable definitions and names and provide known data base designations for those type variables available in the DM's data base. If the model is an APL Model rather than a FORTRAN model, the MIP will notify the user and facilitate an entry to the APL environment where it will carry-out an automatic execution of the APL model. Then, upon completion of execution of the APL work, it will automatically return the user to the DAISY environment.

After obtaining values for all input variables, it executes the particular model and produces a User Summary File (USF) that has listed the names and definitions of the output variables generated, and definitions, names and values of the input variables used to produce that result. In addition, the program will update data base variables as required upon completion of the model run.

To explore the capabilities of the MIP, the reader will be taken through a step by step development process of a fully operational model integrated with the DAISY environment.

2.0 The Model Parameter File (MPF)--the Connecting Link

The first step in the process is to provide a connecting link between the model itself and the interface program that will enable the FORTRAN MIP to determine all the parameters of interest concerning the model such as input and output variable names and definitions, required numeric formats, number and location of the model etc. The MPF provides this link and the standardized structure illustrated below ensures an efficient, straightforward method of accumulating the necessary model information required for the program functioning of the MIP.

2.1 PROGRAM INSERT--a Description

Since PROGRAM RMODEL reads the Model Parameter File in 80 character strings (chosen during implementation based on line display characteristics of a CRT), an efficient method of producing the file is to have a procedure of inputting character strings that will automatically be padded on the right with blanks up to the 80 character maximum. This will then ensure that control characters are encountered at the beginning of a line and the file is searched on a logical line by line basis as it progresses through the file.

2.2 The MPF Elements - Data Insertion

Using FIG A-1's example of a completed Model Parameter File, we start the MPF creation process (NOTE: Unless otherwise stated, a carriage return is executed at the end of each user-inserted line):

STEP 1: Set the remote terminal for both upper and lower case (local procedure manuals should be the guide here, for instance on the Wharton School's GP Associates terminals, the user hits "FUNCTION" key and holding this down, hits "C". On other terminals, the procedure is to type in "SET TTY LC"). Using this option enables the user to insert both the upper case and lower case control characters without subsequent editing that would be required if insertion of the standard upper case character strings were utilized. (Note in our example upper case letters were used throughout the file for file consistency and to highlight the lower case control characters.)

```
^A^S412

^TECONOMIC ORDER QUANTITY

^f1:MOD412.DAT[4010,104]

^UIN GENERAL NOTATION, THIS INVENTORY DECISION MODEL MAY  BE  EXPRESSED
AS  T = F(Q,R), WHERE "T" DENOTES THE TOTAL INVENTORY-RELATED COSTS FOR
A GIVEN PLANNING PERIOD, "Q" IS THE QUANTITY  REPLENISHED  OR  ORDERED,
AND  "R"  IS  THE TIMING OF THE REPLENISHMENT.  THE DECISION IS TO FIND
THE VALUES OF THE DECISION VARIABLES "Q" AND  "R"  WHICH  MINIMIZE  THE
TOTAL COST "T".

^VCOST OF ORDERING PER ORDER (C);
TOTAL ANNUAL REQUIREMENT FOR ITEM (D);
DEMAND RATE PER UNIT OF TIME (E);
CARRYING COST PER UNIT PER YEAR (K);
LEAD TIME PER ORDER (L);

^gF8.2;F5.0;F5.0;I3;F3.0;

^hC;K;

^iI5;F5.2;

^jYES

^kYES

^WECONOMIC ORDER QUANTITY (Q);
REORDER POINT (R);
TOTAL INVENTORY COST (T);

^lF8.0;F5.0;F7.2;

^mQ;R;

^nI5;F5.0;

^oYES

^XDEPARTMENT OF DECISION SCIENCES, WHARTON GRADUATE SCHOOL

^OBROADLY DEFINED, INVENTORY IS  A  STOCK  OF  ANY  ECONOMIC  RESOURCES
REMAINING  IDLE  AT  A  GIVEN POINT OF TIME, INCLUDING SUCH NONPHYSICAL
INVENTORIES AS LEVEL OF CASH, INVENTORY  OF  ACCOUNTS  RECEIVABLE,  AND
INVENTORY  OF  HUMAN  TALENTS  AS  WELL AS MORE FAMILIAR INVENTORIES OF
PHYSICAL GOODS AND MATERIALS.
     AS A SIMPLISTIC EXAMPLE IN THE TASK FORCE ENVIRONMENT, LET:
     1.  THE TOTAL DEMAND OF THE ITEM FOR THE NEXT PERIOD (D),  BE  THE
           ANTICIPATED TOTAL OF FIGHTER PLANES LOST IN THE SEIZURE OF AN
           ISLAND OVER A TEN DAY PERIOD.  (I.E.  D = 30)
```

```
2. THE "CARRYING COST" FOR MAINTAINING ONE UNIT OF INVENTORY
   (PLANE) FOR TEN DAYS INCLUDES FUEL, REPLACEMENT PARTS, AND
   MAINTENANCE COSTS. (I.E. K = $20,000)
3. KNOWN COST OF ORDERING REPLACEMENTS FROM NEAREST CONUS
   TRANSPORTATION POINT INCLUDES COST OF SURFACE/AIR FUEL COST,
   AND PERSONNEL COSTS DURING TRANSPORTATION. (I.E. C =
   $5,000)
4. THERE IS A FOUR DAY LEAD TIME FOR TOTAL TRANSIT TIME FROM U.S.
   TO ARRIVAL OF PLANE ONBOARD THE AIRCRAFT CARRIER. (I.E. L =
   4 DAYS)
5. THE RATE OF CASUALTY ACCUMULATION (USAGE) OVER THE TEN DAY
   PERIOD AVERAGES 3 PLANES PER DAY. (I.E. E = 3)
6. IF USER PROVIDED DATABASE VARIABLES WITH EQUIVALENT AND
   APPLICABLE NAMES (C = DAT301), (K = DAT308) AND TYPED IN
   VALUES FOR D, E, L MENTIONED ABOVE, THEN MODEL 412 WOULD
   PROVIDE THE ANSWERS FOR Q, R, AND T.

^M25.8.75   ;MITCHE

^L
```

Figure A-1
Sample MPF

STEP 2: Activate program INSERT by typing

```
execute insert.f4
```

The program will respond as follows:

```
NAME OF MODEL PARAMETER FILE TO BE CREATED:
```

The user should now enter the name of the parameter file in the format PARNNN.DAT where NNN is the three digit number selected for this model. The numbering conventions required are : 2XX=ALGOL model; 3XX = APL model; 4XX = FORTRAN model. So for our model, the DM would enter:

```
PAR412.DAT
```

PROGRAM INSERT has now created the named file and notifies the user that it is ready to accept the first line of text by displaying:

```
ENTER NEXT SENTENCE:
```

(From this point on, each MPF Section explanation will follow the pattern of (1) Section title, (2) Section description of what is required from user, (3) an example of a typical user insertion.)

Model Number (^S)

The first sentence will now be entered containing the escape control pair ^A signifying the start of the file, followed by ^S indicating the model number follows next

```
^A^S412
```

```
ENTER NEXT SENTENCE:
```

Model Name (^T)

Line 2 contains the control pair (^T) followed by the name of the model

^TEconomic Order Quantity

ENTER NEXT SENTENCE:

Model File (^f)

The next control pair (^f) indicates that the ensuing character string contains the model file location. In our example this location is given by the device number (DEC10 uses "1" to indicate disk storage), file name and extension followed by the user's workspace made up of the Project/ Programmer Number.

^f1:MOD412.DAT[4010,104]

ENTER NEXT SENTENCE:

Model Description (^U)

This next section is the first multi-line character string encountered up to this point. For any multi-line section, the user may type up to and including column 80 prior to a carriage return and continue data entry on succeeding lines. The end of the multi-line section is determined (just as in the single line section) when the MIP encounters the next up-arrow at the start of the next MPF Section. All normal typing conventions (punctuation, capitalization, etc.) are available on any line. The current section of interest is the MODEL DESCRIPTION segment whose purpose is to provide the potential user with a brief statement of purpose of this model in order for the user to assess the possible applicability of the model to the present decision environment.

^UIn general notation...total cost "T".

ENTER NEXT SENTENCE:

Input Variable (^V)

The control pair ^V signifies that the next section contains the generic definitions and names for all input variables required by the user. The variable definition can be up to 40 characters in length and the associated variable name must follow on the same line.

The variable name can be up to ten characters in length, but must be enclosed in parentheses and be followed by a semi-colon. The total number of input variables that may now be specified is 20. But, if this number is found to be too restrictive, the basic Model Interface Program (MIP) may be modified by changing the dimensions of one array in the program (see the Program Coordinator for advice on this matter).

^VCost of Ordering per Order (C);

Total Annual Requirement for Item (D);

ENTER NEXT SENTENCE:

Model Input Variable Format (^g)

In order for the MIP to provide the model with data in the required format, the MPF must have the specific formats required for each variable. It is the present section that provides this data using FORTRAN format specifications. For those unfamiliar with the FORTRAN conventions, a basic summary follows of the data types used by the MIP:

(1) Iw-this indicates a decimal integer wi h "w" a number specifying the maximum field width of of this integer (ex. I3 has possible values of -99 to 100)

(2) Fw.d--this indicates a decimal fixed point number with maximum field width of "w" (which includes the decimal point) and "d" specifies the number of decimal places to the right of the decimal point. (ex.F5.3 can contain a number from -.999 to 9.999).

The format statement may contain up to five characters, but this slight constraint is not of great significance when one considers that the MIP can handle numbers in the range for integers of I1 to I9999 and for real numbers from F10.9 to F99.9. Following each format specification, a semi-colon must be inserted with no intervening blanks. For this reason, do not continue partially completed format statements on succeeding lines. Rather, if it does not fit on the present line completely (including the semi-colon) back up to the previous semi-colon, do a carriage return and start the format on the succeeding line.

A final special case occurs when the model is one in which the number of input variables required varies with each application (such as the number of observations supplied by a user for a particular regression analysis or exponential smoothing model run) rather than a fixed number of variables that require values prior to the execution of the model.

This section should then contain the maximum number of format statements (20) followed by a "M" after the last semi-colon. The MIP will then keep asking for values for input variables only up to the time the user signifies that this was the last input variable value to be supplied for this model run by executing a carriage return without inserting a numeric value.

~gF8.2;F5.0;F5.0;I3;F3.0;

or for a regression analysis,

~gF4.2;F6.3;F8.2;F8.2;F8.2;F8.2;.....F8.2;M

ENTER NEXT SENTENCE:

Type of Model Variables contained in the Data Base (~h)

This section allows the model builder (perhaps in conjunction with the data base administrator or by personal knowledge/experience) to specify those generic input variables that contain the type of information that is available in the data base. For instance if the generic model has an input variable called SPEED which requires the value for the speed of an object, the data base may have values for ship's speed under record element SHPSPD or aircraft maximum speed under FLYSPD etc. The ultimate user is then, in essence, reminded that this "type" of information is available in the data base and, if known to the user or the user has access to a data retrieval system such as WAND [Gerritsen et. al. 1976], the MIP will facilitate retrieval of this information rather than request the value for that variable from the user.

The variables contained in this section, obviously must be a subset of those variables provided in the Input Variable Section and are identified by the same generic names as specified in that section. Again no blanks should be left between entities on the lines and the range of variables would be from 0 to 20 (if all requested data types were available in the data base).

```
    ^hC;K;
ENTER NEXT SENTENCE:
```

Data Base Variable Format (^i)

If the MIP is to affect retrieval of the information and conversion of the data base storage formats to the required generic model format for applicable variables, the specific data base formats must be provided for each variable listed in the previous section. As in the input variable format section, each format must contain no greater than five characters, be followed by a semi-colon, and contain no blanks.

```
    ^iI5;F5.2;
ENTER NEXT SENTENCE:
```

Are data base input variables modified during model run? (^j)

The model builder should indicate whether the execution of this model will cause a modification to the original values supplied from the data base.

^jYes

ENTER NEXT SENTENCE:

Automatically update input data base variables to new values after model run? (^k)

The model builder should indicate (probably in conjunction with the data base administrator) whether this model's modification of input values received from the data base should be reflected in the actual data base values upon completion of the model run.

^kYes

ENTER NEXT SENTENCE:

Output Variables (^W)

Each generic model output variable is listed here--definition and name using the identical procedures outlined under Input Variable Section (^V). Again the maximum number of variables the MIP will presently handle is 20 with the same caveat concerning raising this limit as mentioned in the input section.

^WEconomic Order Quantity (Q);

ENTER NEXT SENTENCE:

Model Output Variable Format (^l)

Identical to the input variable format section, this provides the format of values output by the model upon execution of the model relationships.

^lF8.0;F5.0;F7.2;

ENTER NEXT SENTENCE:

Type of model output variables contained in the data base (^m)

Just as on the input side, these types of variables that are available in the data base are a subset of the generic output variables. The same reminders of identical spelling of names as found in the output variable section as well as omitting blanks and following each name with a semi-c|on are also applicable here.

^mQ;R;

ENTER NEXT SENTENCE:

Data Base Output Variable Format (^n)

This provides the storage format for the type output variables found in the data base as in the input data base format section.

^nI5;F5.2;

ENTER NEXT SENTENCE:

Automatically update output data base variables to new values after model run? (^o)

The model builder (probably in conjunction with the data base administrator) should indicate whether this model's modification of output values received from the data base should be reflected in the actual data base values upon completion of the model run.

^oYes

ENTER NEXT SENTENCE:

Model Source (^X)

Subsequent users may require additional information on the development of the specific relationships of this model. This section provides the development site for this model to indicate where this additional clarification might be obtained.

^XDepartment of Decision Sciences, Wharton Graduate School

ENTER NEXT SENTENCE:

Model Text (^O)

This section contains a detailed description of the mathematical/quantitative relationships underlying the development of this model. It is an amplification of the shorter Model Description section described earlier and is meant to provide a deeper understanding of the critical constraints and relevance of the model's outputs to the problem situation at hand.

^OBroadly defined, inventory is a stock of any economic resources....

ENTER NEXT SENTENCE:

Date of last Model modification and name of documenter (^M)

Entering this data allows the model management system to maintain the latest change status of models resident in the system. As decision makers become more familiar with the system's models, it is certainly expected that the model will not remain a static entity, but evolve to meet the needs of management. The date is day, month , year separated by periods, with a semi-colon following the date and the initials or last name of the person making the modification.

```
^M25.8.75 ;MITCHE
```

ENTER NEXT SENTENCE:

End of File marker (^L)

This terminates the MIP's search of the MPF.

```
^L
```

ENTER NEXT SENTENCE:

Now that we've completed the MPF, we exit from PROGRAM INSERT by doing a carriage return immediately after receipt of this ENTER NEXT SENTENCE: statement. Since PROGRAM INSERT interprets any line that has all blanks from position 1 to 20 as a signal that no more data is to be entered, it then closes the output file and returns the user to the monitor level.

3.0 The Model-MIP Integration

At this point we now have installed the "equalizing function" that allows the common interface of the MIP to access all the models available to DAISY. But the model builder has a number of additional tasks to perform before the MIP will embrace this new model.

3.1 Model Storage Conventions

STEP 3: The next step is to store the model on the disk medium and rename the model file MODNNN.DAT. In this case, based on our example, the name will be MOD412.DAT.

Now we must add a "front end" and a "back end" to the actual model.

3.2 The Model "Front End"

STEP 4: The "front end" is necessary because the MIP gathers the values for the generic input variables and places them into file "IN412.DAT" in the specified format needed by the model. The model must then read these values from the input file. So the "front end" for FORTRAN model 412 will be inserted in file MOD412.DAT as follows: (Note: Comment statements are indicated by C in the leftmost position in steps 4, 5 and 6 of the FORTRAN coding)

```
        CALL RELEAS (1)
C1 Open a disk file "IN412" for sequential reading of its
contents using channel 1.
        CALL IFILE (1,"IN412")
C2 Read the contents of the file using the generic formats
```

for each variable.

```
        READ(1,500,END=700)  C
5500    FORMAT(F8.2)
        READ(1,501,END=700)  D
501     FORMAT(F5.0)
        READ(1,501,END=700)  E
        READ(1,502,END=700)  K
502     FORMAT(I3)
        READ(1,503,END=700)  L
503     FORMAT(F3.0)
```

3.3 The Model "Back End"

STEP 5: Now that it has the required values, the model may proceed with execution, but the model cannot as yet transmit its ultimate output values back to the user in DAISY. This problem is solved by putting on the "back end" to this FORTRAN model by having the model place the output values into an output file which the MIP then reads and passes on to the User's Summary File (USF). The back end appears in MOD412.DAT as:

```
C Release channel 1 from duty in reading from file "IN412"
                CALL RELEAS (1)
C Create a file on disk ("OT412") for sequential entering of
data using channel 1
                CALL OFILE(1,"OT412")
C3 Write the values for each output value sequentially into
the output file using the generic formats for each variable.
```

```
                WRITE(1,600) Q
600             FORMAT(F8.0)
                WRITE(1,601) R
601             FORMAT(F5.0)
                WRITE(1,602) T
602             FORMAT(F7.2)
C Close file "OT412"
                END FILE 1
```

3.4 Model to MIP Linkage

STEP 6: The remaining item to add on the "back end" is the linking mechanism that will activate the last half of the MIP. That is accomplished by inserting a call to a MACRO routine that executes a "RUN" command on the saved file for PROGRAM RETURN,

```
CALL RUNNER (NUMBER)
C NUMBER contains the word "RETURN"
```

The final step in this process is then to "save" the model file by performing the following steps:

```
LOAD MOD412.DAT,RUNNER.MAC
```

The system will respond with,

```
LINK:  Loading
EXIT
```

Then we save this compiled version,

```
SAVE MOD412
```

to which the system confirms the completed process by saying

```
Job Saved
```

(Note, no extension is ever given on the SAVE command)

3.5 The APL Model - A Special Case

At this point we have a fully integrated FORTRAN model, available for the decision maker's use. But this study of the FORTRAN model has left out an increasingly important source of models used in an interactive environment, that of APL models.

To illustrate the similarities between the addition of a model in either system we will use our example model 412 and implement it in APL. All APL models are maintained as functions in the workspace "MODEL". After function MOD312 is added to this workspace, the "front end" and "back end" are accomplished by the creation of the two functions READ312 and PUT312 which perform the same functions as the FORTRAN model:

A. Function MODELNR: obtains the model number 312 from file MODNR.DAT

B. Function OPENFILES: opens files IN312 on channel 2 and file OT312 on channel 3.

C. Function READ312: then performs the reads from file IN312

```
[1]  X<- READAS N

[2]  A<- C <- X

[3]  X<- READAS N

[4]  A<- D <-  X

[5]  X <- READAS N

[6]  A <- E <- X
```

```
[7]  X <- READAS N
[8]  A <- K <- X
[9]  X <- READAS N
[10] A <- L <- X
```

where N was 2 and READAS 2 was a function X <-[<-] [2]N.

D. Function PUT312: finally performed the write out of output values to file OT312,

```
[1] a <- OUT1 <-"F8.0"$(q)
[2] A <- OUT1[->][2]N
[3] A <-OUT2 <-"F5.0"$(R)
[4] A <-OUT2[->][2]N
[5] A <-OUT3 <-"F7.2"$(T)
[6] A <-OUT3 [->][2] N
```

E. Function CLOSEFILES: Closes all files and releases their channels used.

F. Function COMPLETE: Executes a "RUN" command on the saved RETURN file to return to the MIP upon completion of model execution.

So, the wheel has come full circle. Functions B and C parallel FORTRAN's front end and functions D, E and F perform the same functions as the FORTRAN "back end". We have finally reached the point where the Decision Maker (DM) has a functioning model in both APL and FORTRAN. It is now time to utilize these new mathematical modeling tools.

4.0 Executing the Model

4.1 Initiation of the MIP - "RUN MODEL"

STEP 7: The DAISY environment is entered by executing the command:

```
RUN DAISY [400,400]
```

A. The system will respond with the current version of DAISY, the date and the time,

```
DAISY Version 2D(14)-2 on 16-5-76 at 22:12:22
```

B. It will then ask you for the input file of interest for this work session.

```
Input File:
```

C. Let us assume that the DM has been working on a communications plan for the task force with a file name of COMPLN.FIL, so we insert that here,

```
Input File: COMPLN.FIL
```

D. After a carriage return we will receive the DAISY statement,

```
Command:
```

E. At this point we wish to execute Model 412 (EOQ model) since the parts supply problem for electronic equipment has been mounting. To do this we type,

```
RUN MODEL 412
```

4.2 MIP - User Console Interaction

STEP 8: The next thing to appear on the screen will be:

```
MODEL 412: ECONOMIC ORDER QUANTITY
DO YOU WANT TO SKIP TO INSERTION OF VALUES TO GENERIC INPUT
VARIABLES?
```

Option 1: If the reply is "YES", the program will skip directly to the last step prior to model execution and ask the user to supply the appropriate values for each generic input variable. By skipping to data value insertions, the user, is implicitly saying that there are no values available from the data base and the generic definitions and names are suitable for the decision under scrutiny.

Option 2: If the reply is "NO", the user is availing himself of the option of (1) providing data base designations for applicable variables; (2) changing definitions and names of input and output variables to fit the generic model to the specific decision at hand.

Let us say that the DM answers "NO" since he wants to fit the model to the situation at hand.

4.3 Model Data Base Designations

STEP 9: The MIP will respond with,

THERE ARE 4 OUTPUT/INPUT VARIABLE TYPES AVAILABLE FROM THE DATA BASE. DO YOU WANT TO SPECIFY ANY DATA BASE DESIGNATIONS?

Option 1: If the user says "NO", the MIP will skip this section and proceed to the next section which deals with user changes to variable definitions and names. By saying "NO", the user is implying that although there are model variables of the type that are in the data base, he either does not wish to use the data base values or does not know the applicable data base designations

Option 2: If the user says "YES", the MIP will respond with, THERE ARE 2 TYPES OF INPUT VARIABLES AVAILABLE FROM THE DATA BASE:

C ; K ;

VARIABLE TO BE SPECIFIED:

STEP 10: Option 1 -- The user has now been shown the two generic variables that are of the type that are found in the data base. If the user decides that he wants to specify either "C" or "K" he will type in that variable and the next display string will be,

NEXT VARIABLE TO BE SPECIFIED:

Option 2: But, if the DM does not want to specify any input data base designations, he may skip this section by simply executing a carriage return in reply to VARIABLE TO BE SPECIFIED. The MIP will then proceed to the output variable types found in the data base and place the input data base types back on the generic input list to receive their values from the DM later in the program.

STEP 11: At this point if the DM had previously specified "C", he may now add to that selection, the variable "K". But, if he selected "K" in step 10, the MIP has assumed that since the DM skipped over the first variable listed, that variable was not to be designated and the MIP had already removed it from the data base variable list and placed it among the list of variables that would receive its value from the DM later in the program. The natural pattern of selection is then emphasized by using the order displayed criteria of the MIP (analogous to the first-in-first-out, FIFO, basis of inventory control).

STEP 12: After the DM has specified all the input data base types desired, he terminates the process by doing a carriage return following the MIP statement,

NEXT VARIABLE TO BE SPECIFIED:

STEP 13: Following the carriage return, the MIP will list the generic names of all data base type variables, one at a time in the order listed by the user. Assuming that the DM specified "C" and "K" as candidates for data base designations, the MIP will now respond,

C =

and the user may now insert any applicable data base record element/designation up to 10 characters in length.

UNITCOST

Then the next variable specified will appear,

K =

The user then enters the data base designation for "K",

KARRYCOST

A sidenote here is that if after the user made the specification of "C" or "K" and he subsequently forgot the data base designation, the DM can simply do a carriage return following C = or K = and that particular variable will be returned to the generic input list for receipt of a value from the DM later in the MIP.

STEP 14: After the input data base variable types have been examined and received appropriate data base designations, the MIP proceeds to the output data base variable types;

```
THERE ARE 2 TYPES OF OUTPUT VARIABLES AVAILABLE FROM THE DATA
BASE:
Q ;  R ;
VARIABLE TO BE SPECIFIED:
```

With this, the same procedure and restraints delineated for the input variables are applicable here.

4.4 Model Variable Definition and Name Changes

STEP 15: Upon completion of the data base designations section, the DM now has an opportunity to modify generic definitions and names for both input and output variables. In this way a generic model becomes a "specific" model and ultimately, with application of data, a model "instance". The MIP introduces this section by stating:

```
THERE ARE 8 GENERAL INPUT/OUTPUT VARIABLES. DO YOU WANT TO
CHANGE ANY GENERIC DEFINITIONS OR VARIABLE NAMES?
```

OPTION 1: At this juncture if the DM says "NO", the MIP will retain the previously specified data base designation names along with the MPF's generic definition and generic name for non-data base variables and skip to the numeric value insertion stage.

Option 2: If "YES" is answered, the MIP prints out

```
THERE ARE 5 INPUT VARIABLES FOR MODEL 412:
UNITCOST; D       ; E      ;KARRYCOST; L      ;
VARIABLE TO BE CHANGED:
```

The DM may then type in any variable he wishes to modify, again moving from left to right from the first variable listed. Let's say he wanted to modify variable "D". He would type in

"D"

After entering "D", the MIP would type out the appropriate generic definition and name,

TOTAL ANNUAL REQUIREMENT FOR ITEMS (D)

STEP 16:

A. The DM then may change the definition by inserting any character string of up to 40 characters for the new definition followed by a carriage return. But, if he doesn't want to change the definition he just does a carriage return and the generic definition remains intact.

B. Whether or not the DM changes the variable definition, he still has the option to change the variable name. The new variable name may be up to ten characters in length. Again if no change in the name is desired a carriage return will retain the original generic name. For example, the change may appear this way (MIP provided variable definition and name shown in capitals and subsequent user inserted changes in lower case):

TOTAL ANNUAL REQUIREMENT FOR ITEM (D)

Annual Demand Criteria

Demand

C. A special note on the data base designations (UNITCOST, KARRYCOST) now used to identify the former generic names for data base types. Since the data base retrieval program would normally make use of this designation, the name should not be changed once assigned if the user still intends to use this variable to obtain a value from the data base.

D. The DM terminates the input variable change session by executing a carriage return upon receipt of the NEXT VARIABLE TO BE CHANGED: message from the MIP. If he does a carriage return after the initial VARIABLE TO BE CHANGED: is requested, this section is skipped and the user is then in the output variable change phase.

STEP 17: With the completion of the input variable change phase, the output phase comes in with,

```
THERE ARE 3 OUTPUT VARIABLES FOR MODEL 412:
QUEBEC  ; ROMEO  ; T   ;
VARIABLE TO CHANGED:
```

The procedures parallel those carried out with the input variables. Note again the new inclusion of the data base designations for the former generic names of "Q" and "R".

4.5 MIP Data Entry Phase

STEP 18: With the completion of the variable definitions and name changes, the MIP enters the data entry phase. This point would have been reached directly from the beginning of the MIP by answering "YES" to the initial question (STEP 8), in which case every generic input variable would be listed to receive a value from the DM.

But in our example, the DM provided data base designations for two input data base types. So those variables will not be listed for the DM to provide the numeric value, since the value has been obtained from the data base. In addition, any user provided variable name changes will be reflected in this list (such as "DEMAND" below whose generic variable name was originally "D").

```
MODEL 412 INPUT VARIABLES NOT CONTAINED IN THE DATA BASE
REQUIRING DATA VALUE INSERTION BY THE USER:
```

This statement is now followed by the MIP presenting each variable requiring a value from the user. As each variable name is displayed the user inserts a numeric value, either integer or real as the user desires, followed by a carriage return.

In the special case where the model to be executed is one in which the number of input variables required/desired varies with each appllication (such as the number of observations supplied by a user for a particular regression analysis or exponential smoothing model), the user should signify he does not have any further values to enter by executing a carriage return without entering a value for the current variable name displayed. The program will then stop asking for values of input variables and proceed to the model execution phase.

```
DEMAND =
1234
E      =
4444

L        =

55
```

To ensure that each value supplied by the user is acceptable to the model, range checks are performed on each entry. If the value supplied is too low for the model's requirements, the following error message is generated:

```
INPUT VALUE IS LESS THAN MINIMUM PERMITTED OF NNNNNN.  ENTER
NEW VALUE.
```

Conversely, if the value is too large, the following error message is generated,

```
INPUT VALUE EXCEEDS MAXIMUM PERMITTED OF NNNNN FOR THIS
VARIABLE.  ENTER NEW VALUE.
```

Along with these two boundary checks, the program notifies the DM if he inserted a real value when the model required an integer:

```
MODEL VARIABLE REQUIRES INTEGER.  VALUE TRUNCATED TO NNNN.
```

4.6 Return to Monitor Level/DAISY Environment

STEP 19: With the DM being guided through the data insertion phase, the next direct interaction will occur either upon completion of the model run if the model is a FORTRAN model or prior to model insertion if the model is in APL.

If the selected model is contained in the APL environment, the MIP notifies the user by the following message:

```
MODEL 3NN IS WRITTEN IN APL.  FOLLOWING THE NEXT "Command:",
TYPE  "TOAPL"  AND  YOUR TERMINAL TYPE (BIT,TTY,T2741,T4013).
```

ONCE IN APL TYPE ")LOAD MODEL".

Immediately after this message, on the Wharton terminals, the MIP switches the character set from ASCII to APL. Upon completion of the APL model execution, the character set is again switched back to ASCII.

5.0 Model Outputs - The User Summary File

STEP 20: Whether the model is a FORTRAN or an APL model, the MIP will return the user to the monitor level/former environment. In the DAISY environment, the DM receives the result of the model run by typing in the following command:

```
PRINT MODEL OUTPUT 412
```

The User's Summary File will then be printed out on the terminal in the following format:

```
MODEL 412 OUTPUT VALUES:

ECONOMIC ORDER QUANTITY (QUEBEC) = 3333.

REORDER POINT     (ROMEO)    = 3777.

TOTAL INVENTORY COST    (T      ) = 45.00

INPUT VARIABLES USED:

COST OF ORDERING PER ORDER      (UNITCOST) = 1111.00

ANNUAL DEMAND ENVISIONED        (DEMAND)   = 1234.

DEMAND RATE PER UNIT OF TIME    (E      )    =4444.

CARRYING COST PER UNIT PER YEAR (KARRYCOST )   =  444

LEAD TIME PER ORDER             (L      )   =   55.
```

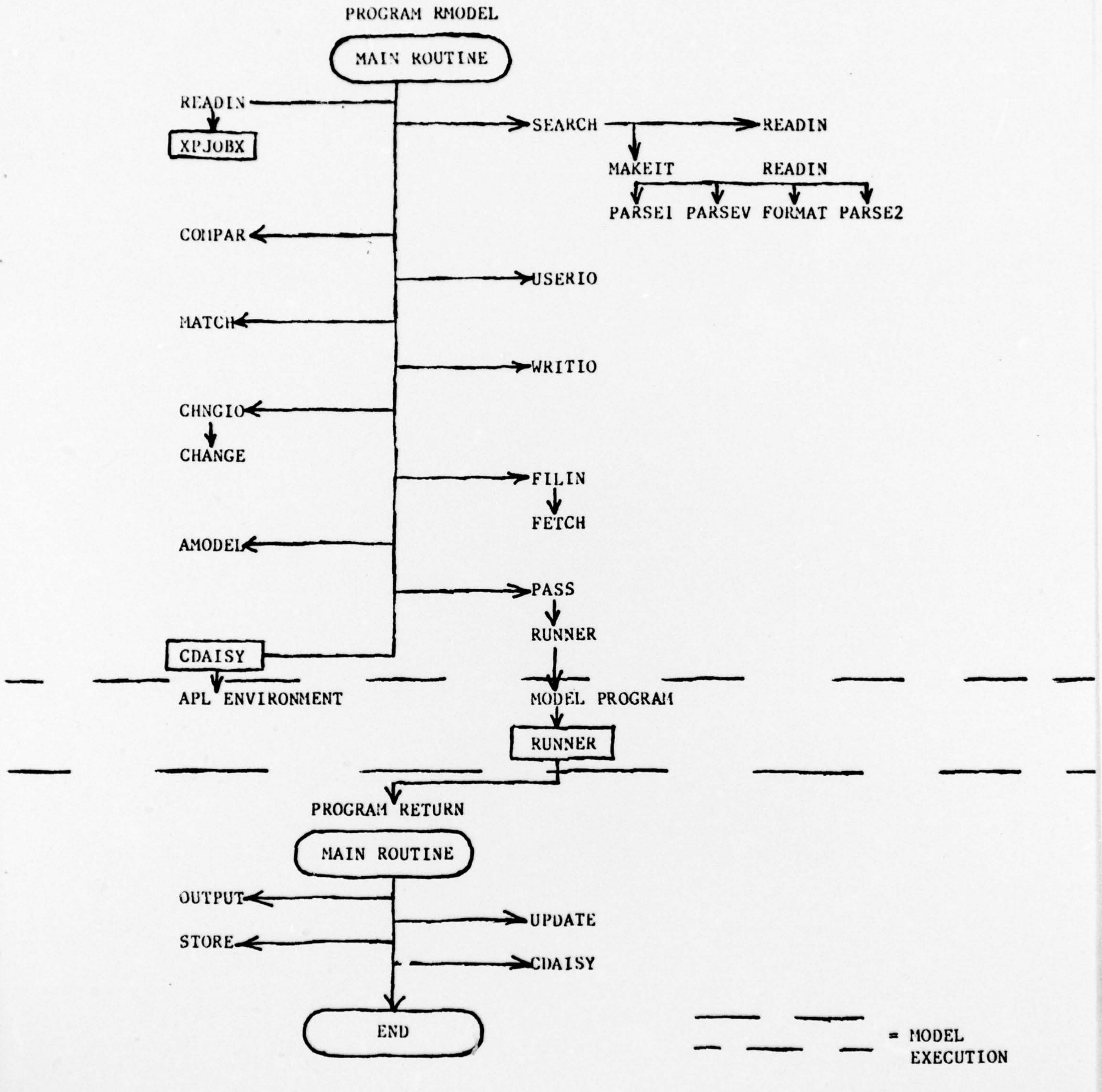
With these 20 steps, the DM has completed the Model Interface Program's "life cycle" from the first introduction of the model to the successful use of the formal decision aid, the mathematical model. As the DM becomes more confident in both his own abilities and that of the MIP in handling these models, the user should be encouraged to actively seek out models that could be of assistance to him in his daily decision processes. The MIP can then become a powerful tool that enables the user to fit the model to the decision at hand rather than fit the problem to the available technique.

Appendix B

Schematic of FORTRAN

Subroutine Linkages for the

Model Interface Program



MACRO FLOW PATTERN

(SUBROUTINES IN FORTRAN AND MACRO)

▭ = MACRO ROUTINE

Appendix C

Subroutine Functional Summary for the Model Interface Program

Appendix C

A. PRE-EXECUTION PHASE (PROGRAM RMODEL)

1. Identify the Model Selected

a. MACRO Routine XPJOBX

PURPOSE: Takes the user's job number (NNN) received when logging on the system and uses this number to identify what file the MPF is contained in (NNNJOB.TMP). The MPF was placed in this file by the POP function bridge from DAISY to RMODEL.

2. Read the MPF

a. Subroutine READIN

PURPOSE: Read next 80 characters of MPF each time the routine is called.

b. Subroutine SEARCH

PURPOSE: Searches through READIN'S 80 character string for an up-arrow and control character of interest. If none is found, it has READIN pull in next 80 characters and continues the search. Once an escape-control pair of interest is found subroutine MAKEIT is called.

c. Subroutine MAKEIT

PURPOSE: Based on the control character pair received from SEARCH, it calls PARSE1, PARSE2, or FORMAT to separate out the applicable information of interest. In addition it computes high and low values permissible for each input variable value as well as calculate the actual number of input variables required and output variables required.

d. Subroutine PARSE1

PURPOSE: This takes the character strings for the input or output generic definitions and places the definitions in one array and then calls PARSEV.

e. Subroutine PARSEV

PURPOSE: Called by PARSE1. This places the names for the generic input or output variables into a separate array for later manipulation.

f. Subroutine PARSE2

PURPOSE: This takes the names of data base variable types and places them in a separate array.

g. Subroutine FORMAT

PURPOSE: This subroutine is called four times by MAKEIT--once each after the generic input, input data base type, output generic, and output data base type variables are parsed into separate arrays. FORMAT then places the specific FORTRAN format for each type variable into separate arrays for later manipulation of the ultimate numeric values.

3. Interaction with DM

a. Subroutine COMPAR

PURPOSE: This produces a 20 by 20 matrix for both input and output variables that has data base variables for rows and generic variables for columns. Initially all 0's, the appropriate subscript receives a "1" when a match occurs between a data base name and a generic name. This then ensures the user will not be asked to supply a value for a variable already contained in the data base.

b. Subroutine USERIO

PURPOSE: The names of the type data base variables (either input or output) are listed on user's terminal and user is asked to type in which variables he desires to supply with specific data base designation. For any variable listed by the program that is not specified by the user as a candidate for a data base designation, the applicable "1" in the check matrix is replaced by a 0 so the user will be asked to supply the value for the variable later in the program.

c. Subroutine MATCH

PURPOSE: The DM is now asked to supply the data base designation for each data base type variable he named in USERIO. If, for some reason he changes his mind, the applicable "1" in the check matrix is zero'd out as done in USERIO.

d. Subroutine WRITIO

PURPOSE: This subroutine now takes the names for the input or output variables (either generic names supplied by the MPF or the new data base descriptions just supplied by the user) and lists them on the DM's terminal.

e. Subroutine CHNGIO

PURPOSE: The DM now may insert the name of any input or output name listed by WRITIO and CHNGIO will then call CHANGE. After return from CHANGE, the DM is asked for the next variable he wants to modify and the process is repeated.

f. Subroutine CHANGE

PURPOSE: For each variable named by the DM in CHNGIO, CHANGE displays on the terminal the generic definition and current variable name. It then accepts any changes the user wants to make on these definitions/names.

4. Obtain Values for Input Variables

a. Subroutine FILIN

PURPOSE: This subroutine first determines the input file name which the model will read to obtain the values of the input variables. (Input file name is INXXX.DAT where XXX is the model number found in the MPF). The input check matrix is then checked for a "1", if none, print out the generic input variable name on screen and user enters a numeric value. It does a range check on this value and if value is invalid, notifies the user to enter another value. If a "1" is in the check matrix, this indicates the variable is found in the data base under the designation provided and subroutine FETCH is called. Whether the value is user supplied or from the data base, the values are placed in the model input file as received with the generic format for that variable.

b. Subroutine FETCH

PURPOSE: This retrieves the value of the data base value in question with specific parameters of storage format and data base designation.

c. Subroutine AMODEL

PURPOSE: This routine determines the name of the file which receives the output values from the model (OTNNN.DAT), the file to receive the User's Summary File (OUTNNN.DAT), and the file to receive parameters created by PROGRAM RMODEL that will be needed by PROGRAM RETURN (CKNNN.DAT) where NNN=model number obtained from MPF.

d. Subroutine PASS

PURPOSE: Pass critical parameters to PROGRAM RETURN needed to complete the accumulation of input and output definitions, names and values in the USF. Then write in file "MODNR" the model number,the parameter holding file name (CKNNN), and the number of DB and NON-DB output/input variables.

e. MACRO Routine RUNNER

PURPOSE: If the model number starts with a 4 (i.e.400, 401, etc.) it is a FORTRAN model and this routine executes a RUN command on the model saved file to start execution of the model. However, if the model number starts with a "3" the model is written in APL and the user is notified that such is the case and to execute the TOAPL DAISY command followed on the same line by the terminal type (T4013, T2740, etc.) the user is on. PASS will also automatically switch to APL characters prior to the return to the DAISY environment. Once in the APL environment type ") LOAD MODEL" and the APL model is executed.

B. Model Execution Phase—PROGRAM MODNNN.DAT

1. Execute the selected model

a. Model "Front End"

PURPOSE: This is not a separate routine but a code written in the model's language as the first thing in the model execution stream. The model simply reads all its input values in the appropriate format from the input file INXXX.DAT.

b. Model "Back End"

PURPOSE: This is not a routine but code written in the model's language that writes out the values for the output values to the output file "OTXXX.DAT".

c. APL execution RUN time instruction:

If model is in APL it executes a "RUN" command to start PROGRAM RETURN.

d. MACRO routine RUNNER

If in FORTRAN the model executes this routine which executes PROGRAM RETURN.

C. Post Execution Phase--PROGRAM RETURN.F4

1. Provide output values/input values to USF

a. MAIN Routine

PURPOSE: This first obtains model number of model just executed from File "MODNR" along with numbers of input and output variables used. It then reads the parameters computed by PROGRAM RMODEL from file CKNNN.DAT.

b. Subroutine OUTPUT:

PURPOSE: This routine reads file OTXXX.DAT for model output values, then reads file INXXX.DAT for model input values used (where XXX=model number). It then places each value with associated input or output variable definition and name in file OUTNNN.DAT (the User's Summary File (USF)).

c. Subroutine UPDATE

PURPOSE: This routine updates the post-execution value for variables, for which the user provided data base designations, by calling routine STORE after converting the format of the output value to the data base storage format.

d. Subroutine STORE

PURPOSE: Data base values are modified following the model execution by placing them in file "DBTG.DAT".

e. MACRO routine CDAISY

PURPOSE: Returns user to the DAISY environment.

Bibliography

Ackoff, Russell, "Management Misinformation Systems", _Management Science_ 14 (December 1967).

Gerritsen, Rob, et. al., "WAND User's Guide", Working Paper 76-01-03, Department of Decision Sciences, Wharton School, University of Pennsylvania, (1976).

Hitch, C. J. and McKean, R. N., _The Economics of Defense in the Nuclear Age_, Harvard, Cambridge, Mass., (1963)

Morgan, Howard L., "Daisy: An Applications Perspective", Working Paper 75-11-03, *Department of Decision Sciences, Wharton School*, University of Pennsylvania, (1975)

Paik, C. M., _Quantitative Methods for Managerial Decisions_, McGraw-Hill Inc., New York, (1973)